DEIM Forum 2011 C7-5

SuperSQL クエリにおけるリンク表現の等価変換

友成　洋介†　　遠山元道††

† 慶應義塾大学理工学部情報工学科　〒 223–8522 神奈川県横浜市港北区日吉 3–14–1
E-mail: †tomonari@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　SuperSQL とは, SQL において SELECT 句の代わりに GENERATE 句を用いて出力媒体を指定し多様なレイアウト表現を可能とする SQL の拡張言語である. 本論文では, SuperSQL クエリの出力媒体を HTML と指定した場合におけるリンク表現の等価変換を行い, 部分更新が可能となるようにクエリを分割するシステムを提案する. そして, 部分更新を可能とした場合の利点を述べる.
キーワード　SuperSQL, HTML, 問い合わせ言語

Equivalent Transformation on Link Expression in a SuperSQL Query

Yosuke TOMONARI† and Motomichi TOYAMA††

† ††Department of Information and Computer Science ,
Keio University
Hiyoshi 3–14–1, Kouhoku-ku, Yokohama-shi, Kanagawa, 223–8522 Japan
E-mail: †tomonari@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

Abstract　SuperSQL is an extension language, which enables us to express a variety of layouts, specifying output medium using GENERATE clause which syntax is "GENERATE< *media* >< *TFE* >" instead of SELECT clause in SQL. This article proposes a system which performs equivalent transformation on a link expression of a SuperSQL query specifying output medium to HTML and divides queries we can update partly. Additionally, we describe effectivenesses of what enables us to update queries partly.
Key words　SuperSQL, HTML, query language

1. はじめに

近年, インターネットは急速に普及し, 一般的な家庭で利用されるようになり, 今までと比べるとより身近なものとなった. その普及とともに WWW では, データベース中の大量のデータを取り扱わなければならなくなった.

遠山研究室で研究されている SuperSQL は, SQL の SELECT 句を GENERATE< *media* >< *TFE* > の構文を持つ GENERATE 句で置き換えたものである. *media* は出力媒体を示し, HTML, PDF, SWF, Excel など様々な媒体を指定する. また *TFE* はターゲットリストの拡張である Target Form Expression を表し, 結合子, 反復子などのレイアウト指定演算子を持つ一種の式である. この SuperSQL を用いることで, 出力媒体に HTML を選択した場合に, Web ページを生成することができる.

出力媒体を HTML と限定した場合, SuperSQL クエリにおけるハイパーリンクを表現する手段は三通りある. *TFE* で深度結合子の%を用いる方法, link 関数を用いる方法, invoke 関数を用いる方法である. %表現の場合, 一つのクエリでリンク元のページとリンク先のページを一括で生成できるが, 一つのクエリであるためこれらのページの一部だけを更新する場合にもそのクエリを再び実行するため, 更新の必要のないページまで再生成しなければならない. そのため Web サイトの規模によりコストが高くなってしまう可能性がある. 一方 link 関数表現では, %表現で記述されていた一つのクエリを分割することにより, 更新の必要な Web ページを限定して更新することが可能となり, 更新にかかるコストを抑えることができる. 本論文では, %表現であったクエリを分割し, link 関数表現に変換するシステムを提案する.

以下, 本稿の構成を示す. まず 2 章で SuperSQL の概要について述べる. 次に 3 章でクエリを分割した場合の有用性を述べ, 4 章で本論文で提案するシステムの手法について述べる. 5 章で評価について述べ, 6 章で結論と今後の課題を述べる.

2. SuperSQL

この章では本論文で対象とする SuperSQL について簡単に述べる. SuperSQL は関係データベースの出力結果を構造化し, 多様なレイアウト表現を可能とする SQL の拡張言語であり, 慶

應義塾大学遠山研究室で開発されている [1] [3]. そのクエリは SQL の SELECT 句を GENERATE< *media* >< *TFE* > の構文を持つ GENERATE 句で置き換えたものである. ここで < *media* > は出力媒体を示し, HTML, PDF などの指定ができる. また < *TFE* > はターゲットリストの拡張である Target Form Expression を表し, 結合子, 反復子などのレイアウト指定演算子を持つ一種の式である.

2.1 結　合　子

結合子はデータベースから得られたデータをどの方向 (次元) に結合するかを指定する演算子であり, 以下の 3 種類がある. 括弧内はクエリ中の演算子を示している.

- 水平結合子 (,)

データを横に結合して出力.

例：Name, Tel

name	tel

- 垂直結合子 (!)

データを縦に結合して出力.

例：Name! Tel

name
tel

- 深度結合子 (%)

データを 3 次元方法へ結合. 出力が HTML ならばリンクとなる.

例：Name % Tel

name	→	tel

%表現で記述されたクエリを SuperSQL システムで実行すると, %演算子の左の属姓値にリンクが貼られた HTML ファイルが生成される. 生成された HTML ファイルコードで, 属性値ごとに飛び先のページを表す HTML ファイル名が埋め込まれる. 元の SuperSQL クエリ名が Qa.sql だとすると, 出力結果の HTML ファイル名は Qa1.html, 飛び先の HTML ファイルの名付け方は, Qa2.html, Qa3.html, ... というようになる.

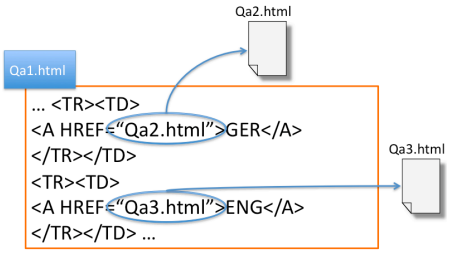

図 1　%表現で生成される html ファイルの例

2.2 反　復　子

反復子は指定する方向に, データベースの値があるだけ繰り返して表示することを指示する. また反復子はただ構造を指定するだけでなく, そのネストの関係によって属性間の関連を指定できる. 例えば

[科目名]! , [学籍番号]! , [評点]!

とした場合には各属性間に関連はなく, 単に各々の一覧が表示されるだけである. 一方, ネストを利用して

[科目名 ! [学籍番号 , 評点]!]!

とした場合には, その科目毎に学籍番号と評点の一覧が表示されるといったように, 属性間の関連が指定される. 以下, その種類について述べる.

- 水平反復子 ([ ],)

データインスタンスがある限り, その属性のデータを横に繰り返し表示する.

例：[Name],

name1	name2	···	name10

- 垂直反復子 ([ ]!)

データインスタンスがある限り, その属性のデータを縦に繰り返し表示する.

例：[Name]!

name1
name2
···
name10

2.3 装　飾　子

SuperSQL では関係データベースより抽出された情報に, 文字サイズ, 文字スタイル, 横幅, 文字色, 背景, 高さ, 位置などの情報を付加できる. これらは装飾演算子 (@) によって指定する.

```
<属性名>@{ <装飾指定> }
```

装飾指定は" 装飾子の名称 = その内容" として指定する. 複数指定するときは各々を", " で区切る.

2.4 関　　数

SuperSQL ではいくつかの関数が用意されている. ここでは代表的な関数を 4 つ紹介する.

2.4.1 imagefile 関数

imagefile 関数を用いると画像を表示することが可能となる. 引数には属性名, 画像ファイルの存在するディレクトリにパスを指定する.

imagefile(id, path="./pic")

2.4.2 link 関数 (出力メディアが HTML の場合のみ)

link 関数は foreach 句と同時に用いる. これらを用いることで深度結合子と同様にリンクを生成することができる.

link(cou.name, file="./menu.sql", att=co.country)

link 関数は第 1 引数に飛び先のページを指定する URL を埋め込まれリンクとなる属性を, 第 2 引数にリンク先のページを生成するクエリファイル名を, 第 3 引数に呼び出されるクエリによって生成されるページを一意的に識別するのに用いる属性を指定する.

foreach 句はリンク先のページを生成するクエリの先頭に記述し, foreach 句で指定された属性ごとにクエリを実行してページを生成する. link 関数の第 3 引数と foreach で指定する属性は一致する必要があり, 且つリンク先を一意に識別出来る属性でなければならない.

foreach 句が記述されたクエリを SuperSQL システムで実行すると, foreach 句で指定されている属性ごとに HTML ファイルを生成する. また link 関数表現で記述されたクエリを Super-

SQL システムで実行して生成される HTML ファイルコードには, link 関数の第 3 引数で指定された属性ごとの URL が埋め込まれる. link 関数で飛び先のページ指定を記述する HTML ファイルを生成し, foreach 句によって foreach 句で指定された属性値ごとのファイルを生成することで, 属性を一意に指定しハイパーリンクを表現することができる.

2.4.3 invoke 関数

invoke 関数はサブクエリを動的に実行し, その生成結果ページへのリンクを生成するための関数である. link 関数の場合, SuperSQL を手動で実行することでリンク先を生成しておくが, invoke 関数の場合, ユーザのクリック動作時に動的にサブクエリを実行し, リンク先を生成する.

invoke(cou.name, file=”./menu.sql”,
condition=”ca.country=”+co.country)

2.4.4 embed 関数

embed 関数を用いることでクエリを分割・合成することが可能になる. 利用方法は別ファイルに保存されたクエリ, もしくは HTML ファイルを埋め込みたい箇所に embed 関数を記述する.

embed(file=”./test.sql” where=”ca.id=” att=ca.id)

3. クエリ分割の利点

深度結合子である%の表現を link 関数に書き換えてクエリを分割する利点を述べる [4].

3.1 web サイトの部分更新が可能

SuperSQL を用いて, GENERATE 句の *media* を HTML と指定して出力した場合, 深度結合子の%表現は, ハイパーリンクとなる. %表現では, Web ページの更新を行う場合, 複数のページをまとめて更新することができる. ところが, この方法では更新の必要のないページまで再生成する. %表現を link 関数表現に変換できれば, 複数更新の必要のあるページだけを選んで更新することが可能になり, 更新する必要のないページを再生成するコストを省くことができる.

3.2 再帰的なリンクの実現

%表現で用いてクエリを記述した場合, 相互リンクを生成するためには同じ記述を何度も繰り返す必要があり, またその繰り返しを行うにしても限界がある. link 関数表現を用いれば, リンク先を指定して再帰的なリンクを実現することが可能である. %表現を link 関数表現に書き換えたクエリの記述を変更する事で, 相互リンクを生成することができる.

3.3 重複の除去

%表現を用いて一つのクエリで Web ページを生成した場合, 同じページを生成してしまう可能性がある. link 関数表現では, 内容の同じページへのリンクを生成するクエリを重複せずに記述することができる.

図 2 は%表現で, 図 3 は link 関数表現でクエリを記述し, Web サイトを生成した結果である. %表現では, 属性によっては重複された結果が出力される場合がある. link 関数表現を用いれば, 一意に識別し, 生成される HTML ファイルの重複を抑制することができる.

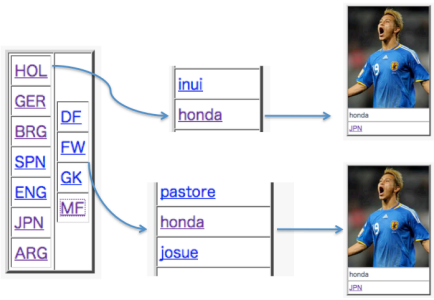

図 2 重複した出力例

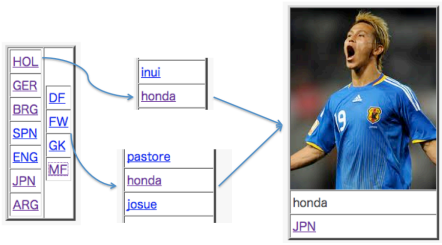

図 3 重複をなくした出力例

3.4 クエリ記述の簡単化

%表現では, リンク元ページの生成とリンク先ページの生成を一つのクエリに記述しなければならない. link 関数表現に変換し, クエリを分割することで, リンク元ページを生成するクエリとリンク先ページを生成するクエリに分けることにより, 一つのクエリの記述を少なくすることができ, クエリを簡単化できる.

4. システム処理の流れ

%表現を link 関数表現に書き換えるシステムを述べる.

%表現で記述されたクエリを分割するために, クエリ中にある TFE 式を木構造にする. 木構造に変換された TFE 式を分割し, 分割した木をクエリに変換することで, クエリを分割する. クエリを分割した後, そのクエリに必要な FROM 句と WHERE 句を定義することで, %表現を link 関数表現への書き換えを完了する [5] [6]. 図 4 はシステムの流れを図示したものである.

図 4 システムの流れ

なお, サンプルとして図 5 にあるクエリを用いて考える.

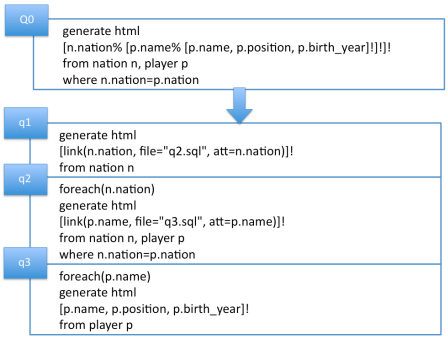

図 5　サンプルクエリ

4.1　木構造を分割

GENERATE 句の TFE 表現を木構造にする．図 6 がサンプルクエリの Q0.sql の TFE 表現を木構造にしたものである．

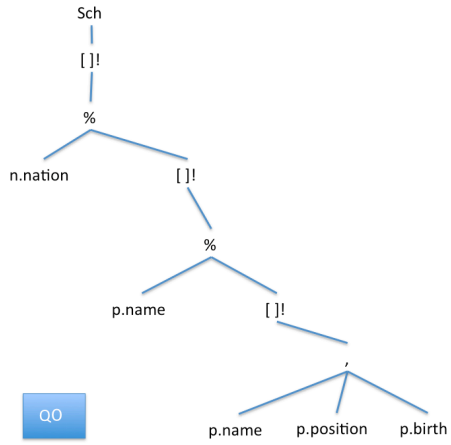

図 6　Q0.sql の木構造

TFE 句の木構造で，%の右側を切り取り分割する．分割する際，分割してできるクエリの個数に応じてクエリにファイル名を与える．%の右側を切り取った後，%表現を link 関数表現に書き換える．%の発見は深さ優先探索で行う．図 7 は Q0.sql を q1.sql, q2.sql, q3.sql に分割したものである．

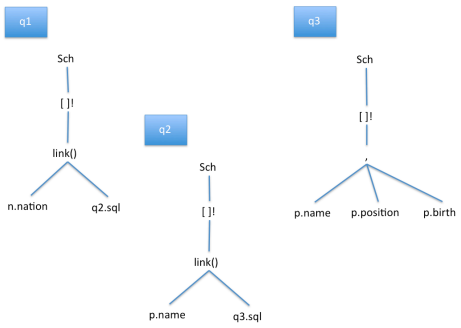

図 7　q1.sql, q2.sql, q3.sql の木構造

4.2　link 関数表現に書き換え

分割した木は%表現にはなっていないだけで，クエリとしての記述ではない．そのため，分割した木を TFE に変換する．分割した木において，中間順による巡回で TFE 式を組み立てる．分割した木構造の葉の左を間順走査する．link 関数の記述がある木の場合，左の属性名を link 関数の第 1 引数とする．このとき，link 関数の第 1 引数を foreach 句の値を保存し，これを link 関数の第 3 引数とする．次に根ノードを走査する．根ノードで link 関数を得る．そして葉の右を走査し，属性を得る．ファイル名を得て，link 関数の第 2 引数とする．そして根ノードを走査する．例えば q1.sql の木を TFE 式に変換する場合，まず左を間順走査し，n.nation を得る．次に根ノードを調査するので link 関数を得る．そして，右を間順走査し q2.sql を得る．

4.3　FROM 句の定義

分割した木構造の葉を一つ一つ探索し，エイリアス名を取得する．元々の FROM 句のエイリアス名と照らし合わせ，必要な FROM 句をクエリに加える．

4.4　WHERE 句の定義

FROM 句の場合と同様に，WHERE 句のエイリアス名を照らし合わせ，必要な WHERE 句をクエリに加える．

4.5　システムによる変換の等価性

本研究で実装したシステムでは，foreach 句で指定した属性は link 関数の第 1 引数と等しいケースのみを扱っているが，一般にはこれらは一致する必要がない．link 関数の第 1 引数の属性の集合を $S(1)$，foreach 句で指定した属性の集合を $S(f)$ とする．link 関数の第 3 引数は飛び先のページを一意に識別できるものでなくてはならないため，$S(1) \supseteq S(f)$ となれば，本研究で実装したシステムを用いて分割する前と後のクエリは等価となる．

クエリ Q の from 句が表集合 T からなることを $Q_f(T)$，where 句が表集合 T からなることを $Q_w(T)$ とする．分割後のクエリを $Q\prime$ とすると，本研究で実装したシステムでクエリを分割した場合は $Q\prime_f(T) \supseteq Q\prime_w(T)$ となる．これを満たす場合において，分割前後のクエリは等価である．

5.　評　　価

5.1　変換の精度

本研究で実装したシステムが，%表現を link 関数表現にどの程度の精度で変換できているかを評価する．%表現で記述されたクエリのサンプルとして，2010 年度のデータモデリングの授業で 14 名の学生が作成したクエリ 14 例を用いた．本研究で実装したシステムでは，SuperSQL クエリにおける装飾子に対応していないため，装飾子を考慮せずに評価を行った．また，学生のサンプルでは link 関数表現がほとんどであったため，その link 関数表現を%表現に本論文著者自身で考え変換した後，本研究で実装したシステムを用いて再び書き換えた．

14 個の%表現で記述されたクエリを link 関数表現に変換した場合における出力の評価の結果が図 8 である．本研究のシステムを用いて，%表現のクエリを link 関数表現に変換した場合

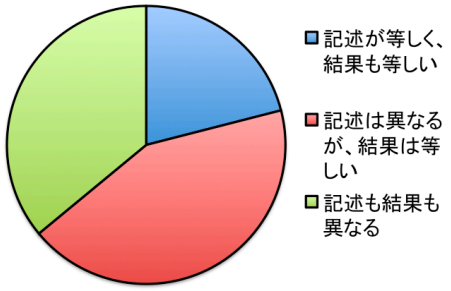

図 8　変換の精度の評価

で, 記述が等しく, またブラウザ上で出力結果も等しくなったクエリは 21%であった. 図 8 の項目の「記述は異なるが結果は等しい」とは, link 関数表現の第 3 引数の値が異なっても, システムで処理した第 3 引数が一意的になっていたので結果が等しくなったと判断できるものである. この場合が 43%であった. 図 9 は変換した記述は異なってしまったが出力結果が等しくなるクエリの例である.

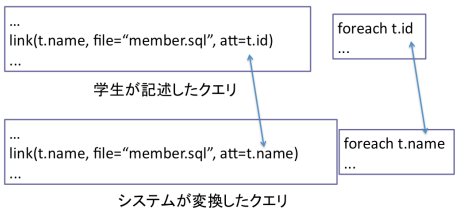

図 9　記述は異なるが結果は等しくなる例

図 9 のクエリを SuperSQL クエリで実行した場合, 国の代表チームの HTML ファイルが生成される. チーム名をクリックすることで, 選手名一覧を表示する web ページに遷移する. この場合, 国名が同じ場合がないため, 結果が等しくなったと判断できる. グラフの「記述は異なるが結果は等しい」の項目は 43%であった. これより, 半分以上は結果が等しくなり, 変換できていることがわかる.

記述も結果も異なった場合の例は図 10 である. 図 10 の上部

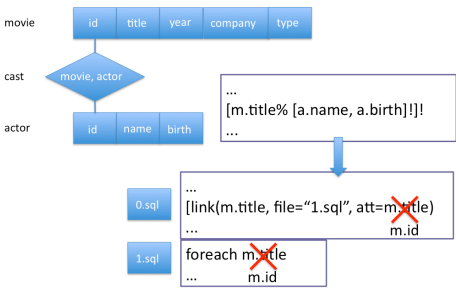

図 10　記述も結果も異なった例

の関係でこのクエリを実行したとき, movie.id=cast.movie と cast.actor=actor.id として映画の出演者を出力する場合, タイトルだけでは一意に判断できないため, 結果が異なってしまった. この link 関数の第 3 引数を主キーに置き換えれば, %表現と link 関数表現の結果が等しくなる.

上記のように link 関数の第 3 引数が主キーの場合が多数あったが, 第 3 引数が主キー以外のクエリもあった. 本研究では, link 関数の第 3 引数を定めてしまったがデータベースの設計や生成する HTML ファイルなどを考慮すると link 関数の第 3 引数を一意に決めることができない. 主キーが多数だからと言っても主キーを第 3 引数に決定してしまってもよいかというと一概には言えない. データベース設計やリンクの飛び先のページ構成を考慮して link 関数の第 3 引数を決定する必要があると考えられる. このことについては次章で述べる.

5.2　手書きによる変換とシステム処理の比較

%表現のクエリを link 関数表現に書き換える際, 本論文著者自身で考えて link 関数に変換する場合と本研究で実装したシステムを用いて変換する場合における処理時間と出力結果の確認時間の比較を行う. SuperSQL クエリの%の個数を 1 個, 2 個, 3 個と変化させ, link 間数表現への書き換えの難易度を変えた場合における時間比較を行った. 自分でクエリを書き換えた場合は, クエリの書き換えにかかった時間と作成して結果が等しいかを確認する時間が含まれている. システムで書き換えた場合は, システムの実行時間と結果が等しいかを確認する時間が含まれている. %のそれぞれの個数に対して 3 個のクエリで書き換えを行い, 3 回の平均時間を測定した.

図 11 が手書きとシステムでの処理の比較のグラフである.

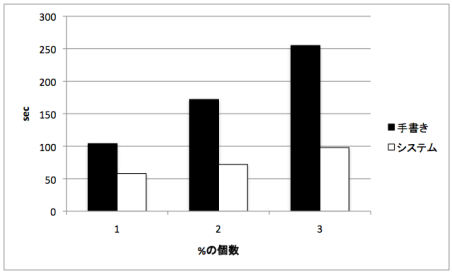

図 11　手書きとシステム処理の確認時間までの比較

グラフから, %の個数が 1 個, 2 個, 3 個の場合いずれも, システム処理の方が約 2 倍ほど早いことがわかる. また, %表現の個数を増やせば増やすだけ, link 関数表現への書き換えに時間がかかる. %表現は構造的にはわかりやすく, link 関数表現は構造が複雑である. 初心者ユーザが SuperSQL を用いる場合, まず%表現からクエリを記述する. ユーザが SuperSQL に慣れるためにクエリを link 関数表現に書き換える際に, 本研究のシステムは利用する価値のあるツールであると考えられる.

6.　結論と今後の課題

6.1　結　　論

本研究では SuperSQL の出力媒体を HTML と指定した場合における%表現であったクエリを, link 関数表現に変換するシステムを実装した. クエリを書き換えた場合における利点を述べた. どの程度の精度でクエリを変換できるかの精度を測定する実験を行ったところ, 60%以上のクエリの出力結果が等価であ

ると判断できるという結果を得た．また, 手書きで%表現を link 関数表現に書き換えた場合と, 本研究で実装したシステムを用いて書き換えた場合の時間を比較したところ, 倍以上の差が出たことがわかった．本研究で実装したシステムは, SuperSQL クエリを初めて用いるユーザが%表現から link 関数表現に書き換える際に利用する価値のあるツールであると考えられる．

6.2 link 関数の第 3 引数誤認問題

今後の課題として, link 関数の第 3 引数の誤認問題が挙げられる．本研究でのシステムは, link 関数の第 1 引数を link 関数の第 3 引数にして処理しているが, 前章でも挙げたように, link 関数の第 3 引数によって出力結果が異なる場合がある．2010 年度の学生が作成したクエリを調査したところ, link 関数の第 3 引数は, 主キーである属性値を使用しているものが多数であった．%表現から link 関数表現に変換し, 出力結果が等しくなる確率を上げるならば, データベースのテーブル上の属性値の中で主キーの属性を link 関数の第 3 引数にすればよい．しかし, その場合でも link 関数の第 3 引数が主キーとなっている場合だけではないため, 全てのクエリを変換することはできない．全てのクエリを%表現から link 関数表現に変換できるようにするためには, 本研究で実装したシステムを利用するユーザが link 関数の第 3 引数を定義する半自動システムでなければならないと考えられる．今後は, link 関数の第 3 引数の誤認問題が解決できるシステムの提案が望まれる．

文　　献

[1] SuperSQL: http://SuperSQL.db.ics.keio.ac.jp/

[2] 遠山　元道：『ターゲットリストの拡張によるデータベース出版と概視の実現』, 信学技報, Vol.93,No.152, P79-88, 電子情報通信学会, 1993

[3] Motomichi Toyama, “SupreSQL: An Extended SQL for Database Publishing and Presentation,” *Preceedings of ACM SIGMOD '98 International Conference on Management of Data*, pp. 584-586, 1998

[4] 石川恭子, 遠山元道: 『データ集約型 Web サイトにおける静的生成コンテンツの部分更新』情報処理学会論文誌, データベース 46(SIG_13(TOD_27)), 1-15, 2005-09-15

[5] T. Seto, T. Nagafuji and M. Toyama, “Generating HTML Sources with TFE Enhanced SQL,” *Proc. ACM Symp. on Applied Computing(SAC '97)*, pp. 96-105, 1997

[6] Motomichi Toyama and Takuhiro Nagafuji: ”Dynamic and Structured Presentation of Database Contents on the Web” Lecture Notes in Computer Science, 1998, Volume 1377/1998, 451-465

[7] E. F. Codd: ”A relational model of data for large shared data banks”, Communications of the ACM, Volume 13 Issue 6, June 1970

[8] Alfred V. Aho, Jeffrey D. Ullman and John E. Hopcroft: ”Data Structures and Algorithms”, Addison Wesley, January 11, 1983

[9] David Gries: ”Compiler Construction for Digital Computers” , John Wiley & Sons, December, 1971

[10] J's GOAL - J リーグ公認ファンサイト: http://www.jsgoal.jp/